

# Der Inhalt (im Überblick)

Intro	xxi
0 Warum Python? Ähnlich und doch anders	1
1 Eintauchen: Sprung ins kalte Wasser	43
2 Listen aus Zahlen: Listendaten verarbeiten	81
3 Listen von Dateien: Funktionen, Module und Dateien	127
4 Formatierte String-Literale: Tabellen aus Daten	177
5 Daten organisieren: Die richtige Datenstruktur	225
6 Eine Web-App erstellen: Webentwicklung	259
7 Bereitstellung: Code überall ausführen	317
8 Mit HTML arbeiten: Web-Scraping	349
9 Mit Daten arbeiten: Datenmanipulation	389
9½ Mit Dataframes arbeiten: Tabellarische Daten	427
10 Datenbanken: Dinge ordnen	451
11 Listenabstraktionen: Datenbankintegrationen	507
12 Bereitstellung in neuem Licht: Der letzte Schliff	571
Anhang: Die zehn wichtigsten Themen, die wir nicht behandelt haben	601
Index	615

# Der Inhalt (jetzt ausführlich)

## Intro

Wir wissen, was Sie denken	xxiii
READ ME	xxviii
Die neueste Python-Version installieren	xxx
Python allein ist nicht genug	xxxii
Konfigurieren Sie VS Code ganz nach Ihrem Geschmack	xxxii
Fügen Sie zwei notwendige Erweiterungen zu VS Code hinzu	xxxvii
Die Python-Unterstützung von VS Code ist auf dem neuesten Stand	xxxiv
Das Team der technischen Sachverständigen	xxxvi
Danksagungen	xxxvii

# 0 Warum Python? Ähnlich und doch anders

Vorbereitungen, Code auszuführen	7
Vorbereitung für Ihre erste Begegnung mit Jupyter	8
Füllen wir den Notebook-Editor mit etwas Code	9
Drücken Sie Shift+Enter, um Ihren Code auszuführen	10
Was ist, wenn Sie mehr als eine Karte ziehen wollen?	15
Ein genauerer Blick auf den Code zum Ziehen einer Karte	17
Die »Großen Vier«: Liste, Tupel, Dictionary und Set	18
Den Kartenstapel mit einem Set modellieren	19
Der »print dir«-Combo-Mambo	20
Hilfe für die Ausgaben von dir	21
Das Set mit Karten füllen	22
Das fühlt sich wie ein Stapel Karten an	24
Was genau ist »card« eigentlich?	25
Suchen Sie etwas?	28
Kurze Pause für eine Bestandsaufnahme	29
Python besitzt eine umfangreiche Standardbibliothek	30
Mit Python schreiben Sie nur den Code, den Sie brauchen	34
Gerade als Sie dachten, Sie seien endlich fertig ...	41

# 1 Eintauchen ***Sprung ins kalte Wasser***

Wie arbeitet der Coach im Moment?	45
Der Coach braucht eine bessere Stoppuhr	46
Bürogespräch	48
Die Datei und die Tabelle sind »verwandt«	51
Aufgabe 1: Daten aus dem Dateinamen extrahieren	52
Ein String ist ein Objekt mit Attributen	53
Daten des Schwimmers aus dem Dateinamen extrahieren	58
Versuchen Sie nicht, zu raten, was eine Methode tut ...	59
Einen String auftrennen (»splitten«)	60
Es gibt noch was zu tun	62
Lesen Sie Fehlermeldungen von unten nach oben	66
Vorsicht beim Kombinieren von Methodenaufrufen	67
Probieren wir es mit einer anderen String-Methode	69
Wir brauchen nur noch ein paar Variablen	72
Aufgabe Nummer 1 ist erledigt!	77
Aufgabe 2: Die Daten in der Datei verarbeiten	78

# 2 Listen aus Zahlen ***Listendaten verarbeiten***

Aufgabe 2: Die Daten in der Datei verarbeiten	82
Holen Sie sich eine Kopie der Daten des Coachs	83
Die open-BIF funktioniert mit Dateien	84
Datei mit with öffnen (und schließen)	85
Variablen werden bei Bedarf dynamisch erstellt	88
Eigentlich brauchen Sie die Daten in der Datei	89
Wir haben die Schwimmer-Daten aus der Datei	91
Der nächste Schritt kommt uns bekannt vor	94
Das vorherige Kapitel zahlt sich aus	97
Einen Zeitstring in einen Zeitwert umwandeln	98
Mit Python zu Hundertstelsekunden	100
Ein kurzer Rückblick auf Pythons for-Schleife	102
Jetzt geht's rund – for-Schleifen gegen while-Schleifen	105
Jetzt läuft es fast von selbst, und Sie machen große Fortschritte!	107
Wir behalten Kopien der konvertierten Werte	108
Eine Liste der Listenmethoden ausgeben	109
Es ist Zeit, den Durchschnitt zu berechnen	114
Den Durchschnittswert in einen Schwimmzeitstring umwandeln	115
Es ist Zeit, die Einzelteile zusammenzufügen	119
Aufgabe 2 hat (endlich) die Ziellinie überquert!	122

# 3

## Listen von Dateien

### **Funktionen, Module und Dateien**

Sie haben den nötigen Code schon fast beisammen	129
Eine Funktion in Python erstellen	130
Speichern Sie Ihren Code, so oft Sie wollen	131
Einfach den Code kopieren reicht nicht	132
Sämtlicher nötiger Code muss kopiert werden	133
Module verwenden, um Code weiterzugeben	140
Erfreuen Sie sich am Glanz der zurückgegebenen Daten	141
Funktionen geben bei Bedarf ein Tupel zurück	143
Holen wir uns eine Liste der Dateinamen des Coachs	149
Zeit für etwas Detektivarbeit ...	150
Was können Sie mit Listen anstellen?	151
Liegt das Problem bei Ihren Daten oder Ihrem Code?	159
Entscheidungen über Entscheidungen	163
Suchen wir den Doppelpunkt »in« dem String	164
Sind Sie auf 60 verarbeitete Dateien gekommen?	171
Der Code für den Coach nimmt langsam Form an ...	172

# 4

## Formatierte String-Literale

### **Tabellen aus Daten**

Einfache Balkendiagramme mit HTML und SVG	182
Von einem einfachen Diagramm zum Balkendiagramm für den Coach	185
Die im HTML benötigten Strings mit Code erstellen	186
Die String-Verkettung skaliert nicht	189
f-Strings sind ein sehr beliebtes Python-Feature	194
Mit f-Strings ist die Erzeugung von SVG ein Kinderspiel!	195
Die Daten sind vollständig, oder nicht?	196
Sicherstellen, dass alle benötigten Daten zurückgegeben werden	197
Die Zahlen sind da, aber sind sie auch benutzbar?	198
Es fehlt nur noch das Ende der Webseite	207
Wie das Lesen aus Dateien klappt auch das Schreiben in Dateien völlig schmerzfrei	208
Es ist Zeit, Ihr Kunstwerk zu präsentieren	211
Jetzt sind nur noch zwei ästhetische Anpassungen nötig ...	212
Eine weitere selbst geschriebene Funktion	214
Erweitern wir das Modul um eine neue Funktion	215
Was ist mit dem Hundertstelwert los?	218
Runden ist nicht das Richtige (jedenfalls nicht in diesem Fall)	219
Es geht gut voran ...	221

# 5

## Daten organisieren *Die richtige Datenstruktur*

Eine Liste mit den Namen der Schwimmer erstellen	227
Der Liste-Set-Liste-Trick	229
Jetzt hat der Coach eine Liste mit Namen	231
Eine kleine Änderung macht einen »großen« Unterschied	232
Jedes Tupel ist einmalig	233
Superschnelle Lookups mit Dictionaries	236
Dictionaries verwenden Schlüssel/Wert-Paare für das Lookup	237
Anatomie eines Dictionary	240
Dictionaries sind für schnelle Lookups optimiert	248
Das gesamte Dictionary ausgeben	249
Das pprint-Modul erstellt einen »Pretty Print« Ihrer Daten	250
Das Dictionary mit den Listen ist leicht zu verarbeiten	251
Langsam nimmt die Sache Gestalt an	252

# 6 Eine Web-App erstellen **Webentwicklung**

Flask aus dem PyPI installieren	261
Den Ordner für die Web-App vorbereiten	262
Bei der Arbeit mit Code haben Sie verschiedene Optionen	265
Anatomie einer Flask-Web-App	267
Schrittweiser Aufbau der Web-App ...	274
Was hat es mit diesem NameError auf sich?	279
Flask unterstützt Session-Verwaltung	281
Flasks Session-Verwaltung benutzt ein Dictionary	282
Den Code mit der »besseren Lösung« reparieren	285
Der Einsatz von Jinja2-Templates spart Zeit	291
base.html erweitern, um weitere Seiten zu erstellen	293
Drop-down-Menüs dynamisch erzeugen	296
Irgendwie müssen die Formulardaten verarbeitet werden	301
Die Formulardaten liegen in einem Dictionary vor	302
Für Funktionsparameter können Standardwerte angegeben werden	307
Standardparameterwerte sind optional	308
Die finale Version Ihres Code, Teil 1 von 2	309
Die finale Version Ihres Code, Teil 2 von 2	310
Für eine erste Web-App sieht das schon ganz gut aus	312
Das System des Coachs ist einsatzbereit	313

# 7 Bereitstellung **Code überall ausführen**

Etwas stimmt immer noch nicht ganz	325
Jinja2 führt den Code zwischen {{ und }} aus	330
Zehn Schritte zur Cloud-Bereitstellung	332
Ein Beginner-Account reicht völlig aus	333
Niemand hält Sie davon ab, einfach loszulegen ...	334
Im Zweifel nutzen Sie die Standardeinstellungen	335
Die Platzhalter-Web-App macht noch nicht viel	336
Eigenen Code auf PythonAnywhere bereitstellen	337
Packen Sie Ihren Code in der Konsole aus	338
Konfigurieren Sie den Web-Tab, damit er auf Ihren Code verweist	339
Die WSGI-Dateien der Web-App anpassen	340
Ihre in der Cloud gehostete Web-App ist bereit!	344

# 8

## Mit HTML arbeiten *Web-Scraping*

Der Coach braucht mehr Daten	350
Machen Sie sich vor dem Scrapeen mit den Daten vertraut	352
Wir brauchen einen Aktionsplan ...	353
Schrittweise Anleitung zum Web-Scraping	354
Zeit für etwas Web-Scraping-Technologie	356
Das rohe HTML-Markup von Wikipedia auslesen	359
Die ausgelesenen Daten untersuchen	360
Slices können aus beliebigen Folgen herausgeschnitten werden	362
Anatomie der Slices, Teil 1 von 3	363
Anatomie der Slices, Teil 2 von 3	364
Anatomie der Slices, Teil 3 von 3	365
Zeit für etwas HTML-Parsing-Power	370
Die »Suppe« nach interessanten Tags durchsuchen	371
Die zurückgegebene »Suppe« ist ebenfalls durchsuchbar	372
Welche Tabelle enthält die gesuchten Daten?	375
Vier große Tabellen und vier Gruppen mit Weltrekorden	377
Jetzt können wir die Daten auslesen	378
Daten aus allen Tabellen extrahieren, Teil 1 von 2	382
Daten aus allen Tabellen extrahieren, Teil 2 von 2	383
Die verschachtelte Schleife war die Lösung!	386

# 9

## Mit Daten arbeiten

### **Datenmanipulation**

Daten Ihrem Willen unterwerfen ...	390
Jetzt haben Sie die nötigen Daten ...	394
Wenden Sie Ihr Wissen an!	396
Haben wir zu viele Daten?	399
Die Staffeldaten ausfiltern	400
Nun können wir unsere Balkendiagramme aktualisieren	401
Python besitzt eine eingebaute JSON-Bibliothek	403
JSON ist textbasiert, aber nicht schön	404
Weiter mit der Web-App-Integration	408
Eine Anpassung und ein Copy-and-paste-Vorgang reichen	409
Die Weltrekorde zum Balkendiagramm hinzufügen	410
Ist Ihre neueste Version der Web-App bereit?	414
PythonAnywhere ist für Sie da ...	418
Auch das Hilfsprogramm muss hochgeladen werden	419
Die neueste Version der Web-App bei PythonAnywhere bereitstellen	420
Den neuesten Code auf PythonAnywhere ausführen	421
Hilfsprogramme vor der Bereitstellung testen	422
Die Aufgabe täglich um 1:00 Uhr morgens ausführen	423

# Mit ~~Elefanten~~ Dataframes arbeiten

## 9½ **Tabellarische Daten**

Der Elefant im Raum ... oder ist es ein Panda?	428
Ein verschachteltes Dictionary mit pandas?	429
Halten Sie sich zunächst an die Konvention	430
Eine Liste mit pandas-Dataframes	431
Spalten aus einem Dataframe auswählen	432
Dataframe zu Dictionary, erster Versuch	433
Unnötige Daten aus einem Dataframe entfernen	434
Den pandas-Bedingungsausdruck verneinen	435
Dataframe zu Dictionary, zweiter Versuch	436
Dataframe zu Dictionary, dritter Versuch	437
Noch ein verschachteltes Dictionary	438
Vergleich zwischen gazpacho und pandas	442
Dies war nur ein winziger Einblick ...	448

# 10 Datenbanken *Dinge ordnen*

Der Coach hat sich gemeldet ...	452
Planung zahlt sich aus ...	455
Schritt 1: Eine Datenbankstruktur festlegen	457
Serviettenstruktur und -daten	459
Das DBcm-Modul von PyPI installieren	460
Einstieg in DBcm und SQLite	461
DBcm und »with« als Team	462
Benutzen Sie dreifach doppelte Anführungszeichen für Ihren SQL-Code	464
Nicht jede SQL-Anweisung gibt etwas zurück	466
Ihre Tabellen sind bereit (und Schritt 1 ist erledigt)	471
Welche Schwimmer-Dateien brauchen wir?	472
Schritt 2: Die Datenbanktabelle mit Inhalt füllen	473
Sicherheit durch Python's SQL-Platzhalter	475
Wiederholen wir dieses Vorgehen für die Ereignisse	490
Jetzt fehlt nur noch die times-Tabelle ...	494
Die Zeiten stehen in den Schwimmer-Dateien ...	495
Ein Hilfsprogramm zur Datenbankaktualisierung, 1 von 2	501
Ein Hilfsprogramm zur Datenbankaktualisierung, 2 von 2	502
Schritt 2 ist (endlich) abgeschlossen	503

# 11 Listenabstraktionen **Datenbankintegrationen**

Testen wir die Abfragen in einem neuen Notebook	510
Fünf Codezeilen werden zu einer	513
Combo-Mambo ohne Dunder	515
Eine Abfrage erledigt, drei fehlen noch ...	520
Zwei Abfragen erledigt, zwei fehlen noch ...	522
Zu guter Letzt, die letzte Abfrage ...	523
Der Code für die Datenbankhilfsfunktionen, Teil 1 von 2	528
Der Code für die Datenbankhilfsfunktionen, Teil 2 von 2	529
Wir sind fast bereit für die Datenbankintegration	532
Es ist Zeit, den Datenbankcode zu integrieren!	540
Was ist mit dem Template los?	548
Eine Liste der Ereignisse anzeigen ...	552
Jetzt brauchen wir nur noch ein Balkendiagramm ...	556
Überprüfung des aktuellen Codes in <code>swimclub.py</code>	558
Begegnung mit dem SVG-erzeugenden <code>Jinja2-Template</code>	560
Das Modul <code>convert_utils</code>	562
<code>list zip ... wie bitte?!!</code>	565
Ihre Datenbankintegrationen sind fertig!	567

# 12

## Bereitstellung in neuem Licht

### **Der letzte Schliff**

Migration zu MariaDB	575
Die Daten des Coachs zu MariaDB umziehen	576
Drei Anpassungen für schema.sql	577
Die Tabellen wiederverwenden, Teil 2 von 2	578
Überprüfen, ob die Tabellen korrekt eingerichtet wurden	579
Die vorhandenen Daten zu MariaDB übertragen	580
Die Abfragen mit MariaDB kompatibel machen	582
Die Datenbankhilfsfunktionen müssen ebenfalls angepasst werden	583
Eine neue Datenbank auf PythonAnywhere anlegen	586
Das Dictionary mit Datenbankinformationen anpassen	587
Alles in die Cloud kopieren	588
Die Web-App mit dem neuesten Code aktualisieren	589
Nur noch wenige Schritte ...	590
Die Cloud-Datenbank mit Daten füllen	591
Zeit für eine PythonAnywhere-Probefahrt	592
Stimmt mit PythonAnywhere etwas nicht?	594
Der Coach ist überglücklich!	596

# Anhang

## ***Die zehn wichtigsten Dinge, die wir nicht behandelt haben***

1. Klassen	602
2. Ausnahmen (Exceptions)	605
3. Tests	606
4. Der Walross-Operator	607
5. Wo ist switch? Welcher switch?	608
6. Fortgeschrittene Sprachmerkmale	609
7. Nebenläufigkeit	610
8. Typhinweise	611
9. Virtuelle Umgebungen	612
10. Werkzeuge	613